Fayoum University
Fayoum Faculty Of Engineering
Communication & Electronics Department

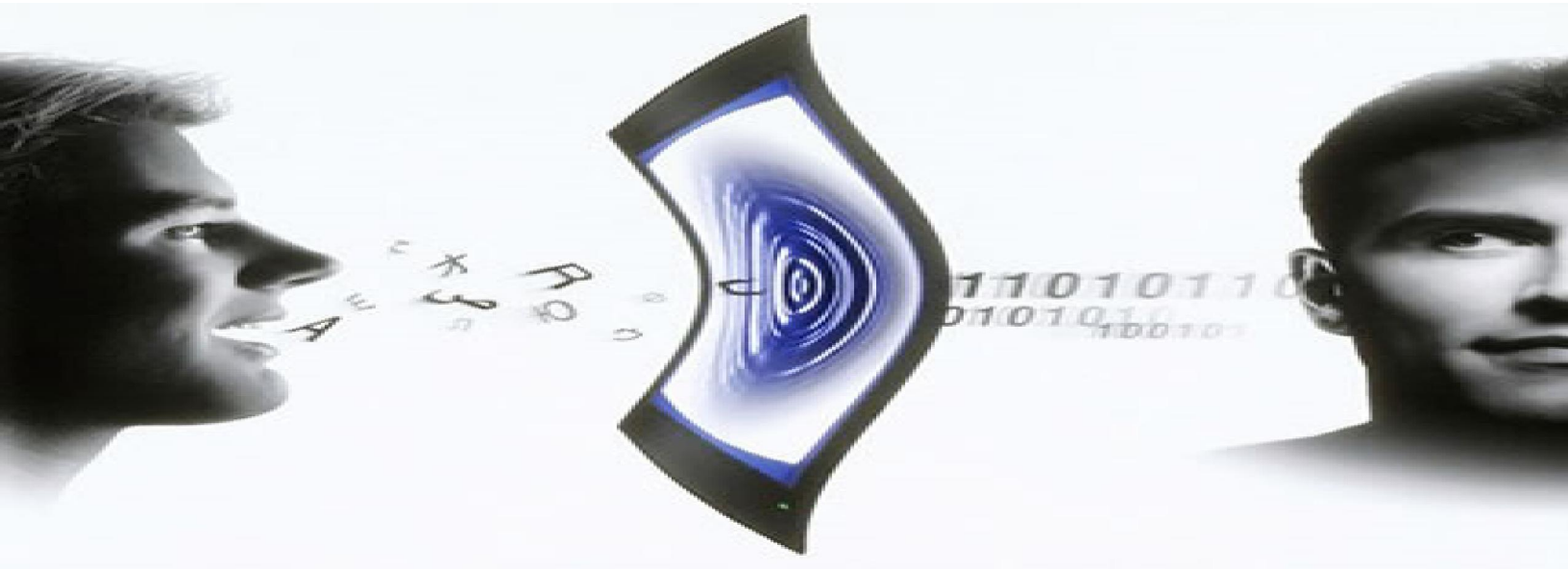
Graduation Project

Voice Over Internet Protocol (VOIP) over WIFI

Supervisor

DR\ Tamer barakat

2013-2014



VOIP (VOICE OVER IP)
over WiFI

By…..

MOSTAFA ISMAIL ABD EL-ZAHER

AHMED EWIES RAGAB

AHMED ABD EL-MENAEM

GOMAA HUSSIEN

MOSTAFA ADEL MANSOUR

A Graduation Project Report Submitted to
the Faculty of Engineering at Fayoum University
in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science
in
" Electronics and Communications Engineering "

Faculty of Engineering, Fayoum University

Under Supervision by :

Dr.Tamer Barakat,
Associate professor

Fayoum, Egypt

July 2013

ABSTRACT

”The Internet does not recognize state and country borders ” .

With VoIP technology, the distinction between local, long-distance, and international calling largely disappears, and callers can save on long-distance and international charges.

VoIP is the family of technologies that allows IP networks to be used for voice applications, such as telephony, voice instant messaging, and teleconferencing. VoIP defines a way to carry voice calls over an IP network, including the digitization and packetization of the voice streams. IP Telephony VoIP standards create a telephony system where higher-level features such as advanced call routing, voice mail, and contact centers can be utilized.

VoIP makes calls significantly **cheaper**. And for a low cost monthly subscription local and national calls can be made for free, and international calls made at a significantly lower rate.

When you subscribe to a VoIP service it is possible to get a phone number for life. You will be able to take your number with you whenever you move (or even travel) with obvious benefits.

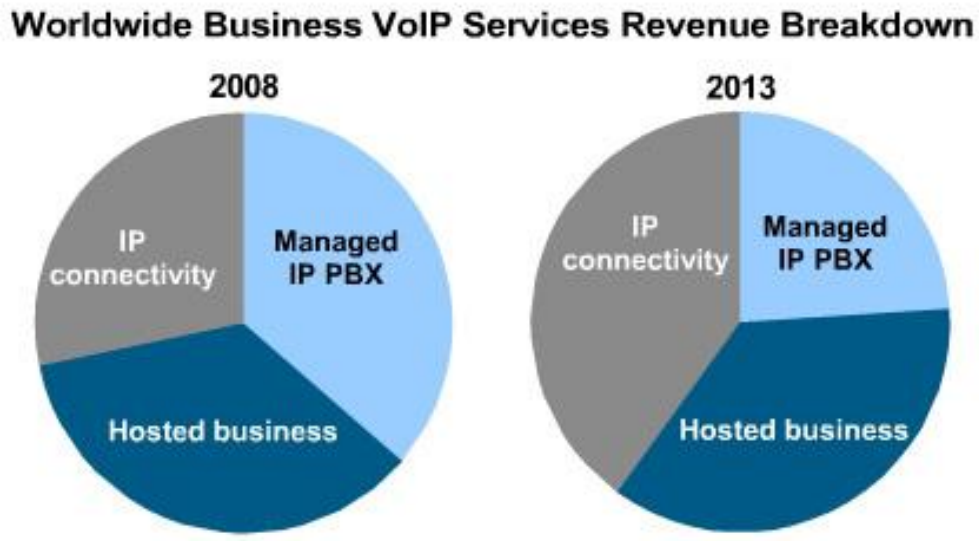
The contract packages that companies who are offering the service contain for the most part all the features and more that your current phone service supplier offers. You are likely to find most VoIP companies offering free voicemail, call forwarding, caller ID, call waiting, call waiting ID, 3 way calling, speed dialing and much more.

VoIP is very much in it's **infancy**; the technology is progressing all the time. Future benefits that we may not be able to envisage yet are certain to appear, you

current analogue phone system will not be able to compete .

Some of the most common value-added features and services for users are:

- Unified messaging.
- Interactive voice recognition.
- Call center administration.
- Voice mail.
- Conferencing services.
- Database queries (for example, e-mail look-ups).
- Customer relationship management.
- Instant messaging and Web browsing.



© Infonetics Research, *VoIP and UC Services and Subscribers Biannual Worldwide and Regional Market Size, Share, and Forecasts*

Index

Problem statement

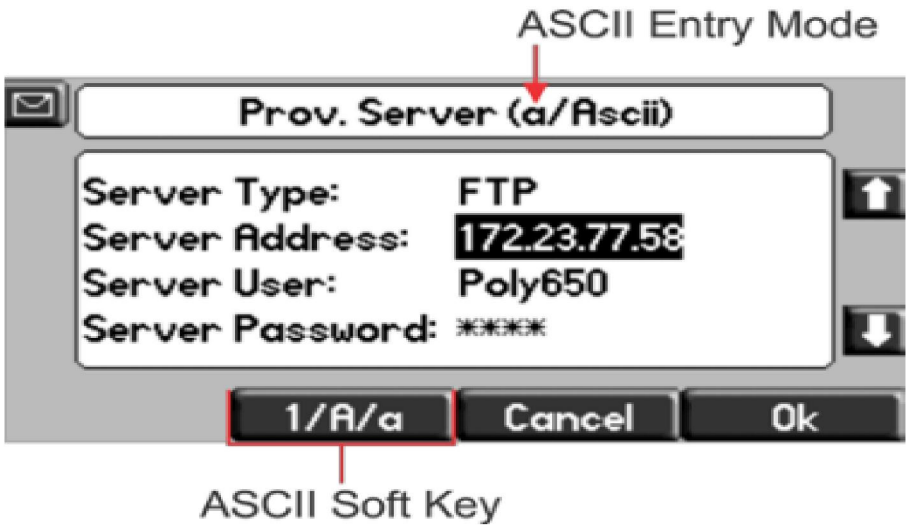
In this part we will discuss the problem that We encountered during the work on the project .

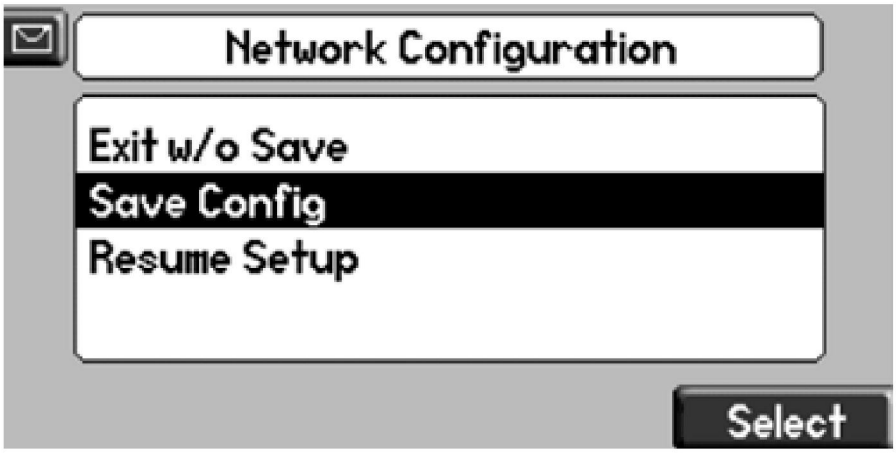
The most important of these was the most difficult problems is :

- **upload a firmware to the ip phones**, and that take along time to be solved .
  to upload a firmware to my ip phone we do the following :
  1. download the Appropriate firmware of my ip phone version that can be downloaded from website of the company.
  2. set a TFTP server on your pc " server " by using a programs such ' **WIN AGENT TFTP SERVER** ' ,**'SOLAR WIND'** . Create a root FTP directory on the computer with full read and write access to all directories and files .
  3. set up the configuration of the phone to enable the phone to connect to the pc by the following :
     - On the phone, press **Menu** > **Settings** > **Advanced** and enter the phone's password. The default is **456**.
     - Choose **Admin Settings** > **Network Configuration** > **Provisioning Server**, and press the Select soft key.
     - Scroll to a field, press the **Edit** soft key, and enter the FTP provisioning boot server information—**in this example**

your computer's IP address—user name, and password.

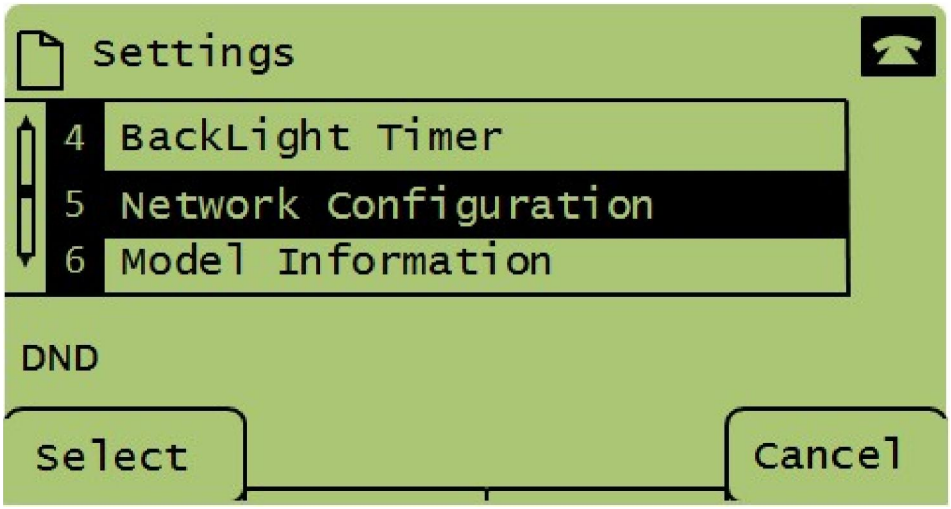
- When finished, press the **OK** soft key, and the Back soft key once or twice. A prompt screen will display.
- Choose **Save Config** and press the Select soft key to **reboot** the phone and save the new configuration.
  You can now place and receive calls.



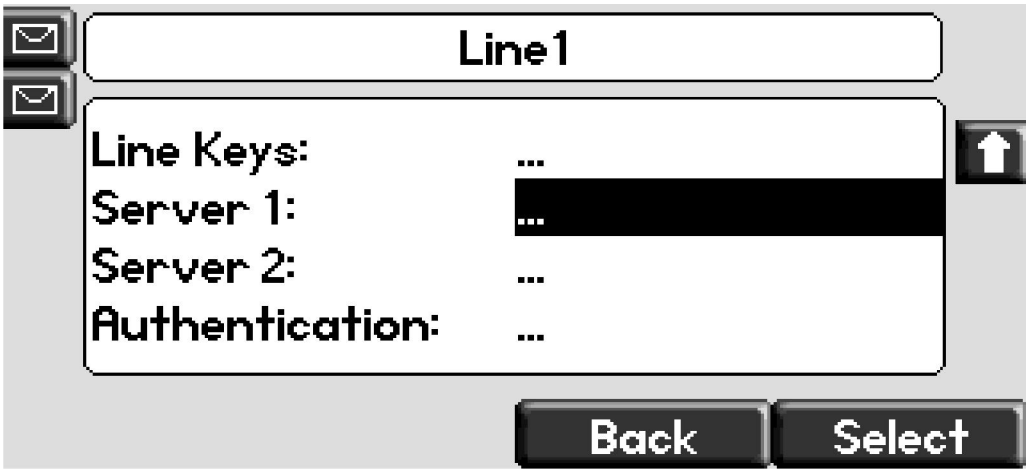


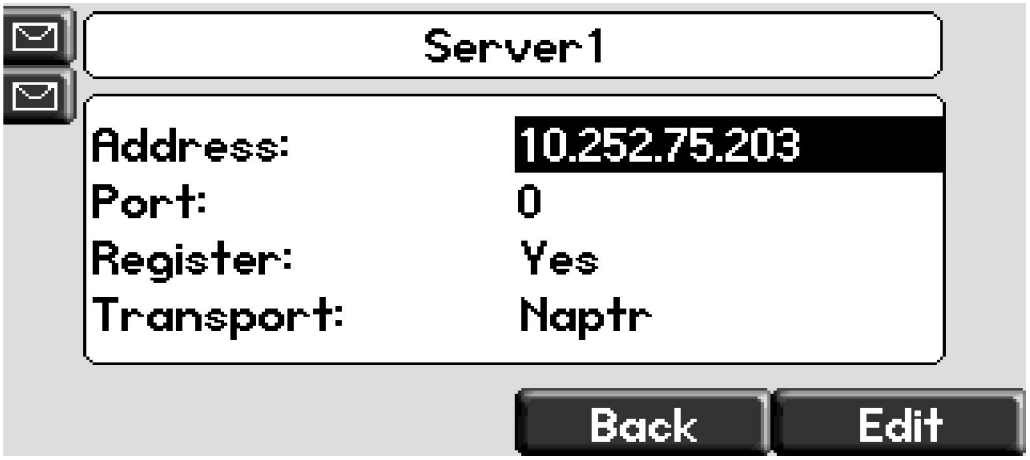
- after doing Previous instructions , **we still can't place or receive calls** due to lines are **not registered** yet because ip phone doesn't have an IP yet beside the call manager express hasn't recognize the two ip phones .

we can set IP for the ip phones as following :



- **configure ip phones to can recognize ‘elastix ‘ call manager express ‘** . that can be done by the following :





- **configure ‘elastix ‘ call manager express ‘ to recognize ip phones** …. illustrated in chapter 5 …

- **configure ‘elastix ‘ call manager express ‘ to support “ voice mail “ , “ conference “ , “ transfer call “ , “ IM “ , “ music on hold “ ,“call back “**… illustrated in chapter 5 …

- **how to make a call between an ip phone and a PSTN phone** , that can be done by configure **“ AN VOIP ADAPTER(ATA)**

that can set an ip address for the PSTN phone and convert calls between ip phone and PSTN phone.

A VoIP Adapter, also known as an Analog Telephone Adapter (**ATA**) will convert a VoIP signal to an analog tone so that you can use existing analog devices such as your phone or fax machine with VoIP service. This allows you all the awesome cost-savings of VoIP without having to replace your existing analog equipment.

Code of the encryption security :

```
1 package sun.security.rsa;
2
3 import java.math.BigInteger;
4 import java.util.*;
5
6 import java.security.SecureRandom;
7 import java.security.interfaces.*;
8
9 import javax.crypto.BadPaddingException;
10
11 import sun.security.jca.JCAUtil;
12
13 public final class RSACore {
14
15     private RSACore() {
16         // empty
17     }
18     public static int getByteLength(BigInteger b) {
19         int n = b.bitLength();
20         return (n + 7) >> 3;
21     }
22     public static int getByteLength(RSAKey key) {
23         return getByteLength(key.getModulus());
24     }
25
26      // temporary, used by RSACipher and RSAPadding.
Move this somewhere else .
27       public static byte[] convert(byte[] b, int ofs,
int len)
     {
28         if ((ofs == 0) && (len == b.length)) {
29             return b;
30         } else {
31             byte[] t = new byte[len];
```

```
32              System.arraycopy(b, ofs, t, 0, len);
33              return t;
34          }
35      }
36      public static byte[] rsa(byte[] msg, RSAPublicKey
key)
37              throws BadPaddingException {
38          return crypt(msg, key.getModulus(),
key.getPublicExponent());
39      }
40      public static byte[] rsa(byte[] msg, RSAPrivateKey
key)
41              throws BadPaddingException {
42          if (key instanceof RSAPrivateCrtKey) {
43              return crtCrypt(msg,
(RSAPrivateCrtKey)key);
44          } else {
45 return crypt(msg, key.getModulus(),
     key.getPrivateExponent());
46          }
47      }
48      private static byte[] crypt(byte[] msg, BigInteger
n, BigInteger exp)
49          throws BadPaddingException {
50          BigInteger m = parseMsg(msg, n);
51          BigInteger c = m.modPow(exp, n);
52          return toByteArray(c, getByteLength(n));
53      }
54      private static byte[] crtCrypt(byte[] msg,
RSAPrivateCrtKey key)
55              throws BadPaddingException {
56          BigInteger n = key.getModulus();
57          BigInteger c = parseMsg(msg, n);
58          BigInteger p = key.getPrimeP();
59          BigInteger q = key.getPrimeQ();
60          BigInteger dP = key.getPrimeExponentP();
61          BigInteger dQ = key.getPrimeExponentQ();
62          BigInteger qInv = key.getCrtCoefficient();
63
64          BlindingParameters params;
65          if (ENABLE_BLINDING) {
66              params = getBlindingParameters(key);
```

```
67              c = c.multiply(params.re).mod(n);
68          } else {
69              params = null;
70          }
71
72          // m1 = c ^ dP mod p
73          BigInteger m1 = c.modPow(dP, p);
74          // m2 = c ^ dQ mod q
75          BigInteger m2 = c.modPow(dQ, q);
76
77          // h = (m1 - m2) * qInv mod p
78          BigInteger mtmp = m1.subtract(m2);
79          if (mtmp.signum() < 0) {
80              mtmp = mtmp.add(p);
81          }
82          BigInteger h = mtmp.multiply(qInv).mod(p);
83
84          // m = m2 + q * h
85          BigInteger m = h.multiply(q).add(m2);
86
87          if (params != null) {
88              m = m.multiply(params.rInv).mod(n);
89          }
90
91          return toByteArray(m, getByteLength(n));
92      }
93      private static BigInteger parseMsg(byte[] msg,
BigInteger n)
94              throws BadPaddingException {
95          BigInteger m = new BigInteger(1, msg);
96          if (m.compareTo(n) >= 0) {
97              throw new BadPaddingException("Message is
larger than                 modulus");
98          }
99          return m;
100     }
101     private static byte[] toByteArray(BigInteger bi,
int len) {
102          byte[] b = bi.toByteArray();
103          int n = b.length;
104          if (n == len) {
105              return b;
```

```
106         }
107         // BigInteger prefixed a 0x00 byte for 2's
complement form,           remove it
108         if ((n == len + 1) && (b[0] == 0)) {
109             byte[] t = new byte[len];
110             System.arraycopy(b, 1, t, 0, len);
111             return t;
112         }
113         // must be smaller
114         assert (n < len);
115         byte[] t = new byte[len];
116         System.arraycopy(b, 0, t, (len - n), n);
117         return t;
118     }
119
120     // globally enable/disable use of blinding
121     private final static boolean ENABLE_BLINDING =
true;
123
124     // maximum number of times that we will use a set
of blinding  parameters
125     // value suggested by Paul Kocher (quoted by NSS)
126     private final static int BLINDING_MAX_REUSE = 50;
127
128     // cache for blinding parameters.
     Map<BigInteger,BlindingParameters>
129     // use a weak hashmap so that cached values are
automatically          cleared
130     // when the modulus is GC'ed
131     private final static
Map<BigInteger,BlindingParameters>
     blindingCache =
132      new WeakHashMap<BigInteger,BlindingParameters>();
133     private static final class BlindingParameters {
134         // e (RSA public exponent)
135         final BigInteger e;
136         // r ^ e mod n
137         final BigInteger re;
138         // inverse of r mod n
139         final BigInteger rInv;
140         // how many more times this parameter object
can be used
```

```
141         private volatile int remainingUses;
142         BlindingParameters(BigInteger e, BigInteger
re, BigInteger             rInv) {
143             this.e = e;
144             this.re = re;
145             this.rInv = rInv;
146             // initialize remaining uses, subtract
current use now
147             remainingUses = BLINDING_MAX_REUSE - 1;
148         }
149         boolean valid(BigInteger e) {
150             int k = remainingUses--;
151             return (k > 0) && this.e.equals(e);
152         }
153     }
154     private static BlindingParameters
getBlindingParameters
155             (RSAPrivateCrtKey key) {
156         BigInteger modulus = key.getModulus();
157         BigInteger e = key.getPublicExponent();
158         BlindingParameters params;
159        // we release the lock between get() and put()
160         // that means threads might concurrently
generate new blinding
161         // parameters for the same modulus. this is
only aslight waste
162         // of cycles and seems preferable in terms of
scalability
163         // to locking out all threads while generating
new parameters
164         synchronized (blindingCache) {
165             params = blindingCache.get(modulus);
166         }
167         if ((params != null) && params.valid(e)) {
168             return params;
169         }
170         int len = modulus.bitLength();
171         SecureRandom random =
JCAUtil.getSecureRandom();
172         BigInteger r = new BigInteger(len,
random).mod(modulus);
173         BigInteger re = r.modPow(e, modulus);
```

```
174         BigInteger rInv = r.modInverse(modulus);
175         params = new BlindingParameters(e, re, rInv);
176         synchronized (blindingCache) {
177             blindingCache.put(modulus, params);
178         }
179         return params;
180     }
181
182}
```